The modified matrix splitting iteration method for computing PageRank problem

Zhaolu Tian[a], Xiaoyan Liu[b,*], Yudong Wang[c], P.H. Wen[d]

[a] *School of Applied Mathematics, Shanxi University of Finance and Economics, Taiyuan 030006, P.R.China*

[b] *College of Data Science, Taiyuan University of Technology, Taiyuan 030024, P.R.China*

[c] *College of Textile Engineering, Taiyuan University of Technology, Taiyuan 030024, P.R.China*

[d] *School of Engineering and Materials Science, Queen Mary University of London, London E14NS, UK*

Abstract

In this paper, based on the iteration methods [3,10], we propose a modified multi-step power-inner-outer (MMPIO) iteration method for solving the PageRank problem. In the MMPIO iteration method, we use the multi-step matrix splitting iterations instead of the power method, and combine with the inner-outer iteration [24]. The convergence of the MMPIO iteration method is analyzed in detail, and some comparison results are also given. Several numerical examples are presented to illustrate the effectiveness of the proposed algorithm.

Key words: *PageRank; Inner-outer iteration; Convergence; Matrix splitting; Power method*

1. Introduction

With the booming advance of the Internet and its technology, web search engines have become the most popular Internet tools to retrieve information. Google's PageRank is one of the most important algorithms to determine the importance of Web pages.

In the PageRank problem, the main work is to compute the PageRank vector x^*, which is a probability vector, i.e.,$\|x^*\|_1 = 1$, and satisfies the following relation:

$$\mathcal{A}x^* = [\alpha P + (1-\alpha)ve^T]x^* = x^*, \tag{1.1}$$

where $\alpha \in (0,1)$ is the damping vector, $P \in R^{n\times n}$ is a column-stochastic matrix, e is a column vector of all ones, and v is called the personalization or the teleportation vector. The PageRank problem (1.1) often arises in web ranking[1,4,6,8], for example, the hyperlink structure of the web and modeling the graph by the Markov chain, etc.

In the last decades, many numerical methods have been proposed for computing the PageRank vector. However, due to the huge size and sparsity of the matrix $\mathcal{A}$, the fast eigenvector solvers derived from matrix inversions or decompositions are expensive and prohibitive for computing PageRank vector, then the iteration methods based on matrix-vector product have attracted more and more attention. The power method [5,12] is the original method used for solving the PageRank problem (1.1). The extrapolation methods [9,11,13,16] were constructed to accelerate the power method when the largest eigenvalue is not well separated from the second one. Recently,

*Corresponding author.

E-mail address: lucyyanxiao@163.com, tianzhaolu2004@126.com.

many Krylov subspace methods have also been proposed to solve the PageRank problem (1.1). The authors [4] gave an Arnoldi-type algorithm, which is a variant of the restarted Arnoldi method [23]. Combined the power method with the thick restarted Arnoldi algorithm [19], the Power-Arnoldi algorithm [26] was presented. A Ritz-value-based Arnoldi-Extrapolation algorithm [28] was developed, which periodically knits extrapolation method with the Arnoldi-type algorithm. In order to modify the inner-outer method, Gu and Wang gave an Arnoldi-Inout algorithm [18] by using the thick restarted Arnoldi method.

Note that the PageRank problem (1.1) can be rewritten as the following linear system [17,21,24,25]:

$$(I - \alpha P)x = (1 - \alpha)v, \tag{1.2}$$

where I is an $n \times n$ identity matrix. Based on the matrix splitting $I - \alpha P = (I - \beta P) - (\alpha - \beta)P$, where $0 < \beta < \alpha$, an inner-outer iteration method [24] was proposed for solving (1.2). Since the matrix $I - \alpha P$ is a nonsingular $M-$matrix [2,20], a class of splitting iteration methods were presented in [7], which are obtained from the M-splittings of the matrix $I - \alpha P$.

Recently, Wen et al. [3] developed a multi-step power-inner-outer (MPIO) iteration method, which is a variant of the PIO iteration [10] by combining multi-step power method with the inner-outer iteration [24]. In this paper, we propose an MMPIO iteration method, which is based on the multi-step matrix splitting iterations instead of the power method and the inner-outer iteration [24]. Finally, several numerical examples are used to show the efficiency of the proposed MMPIO iteration.

The remainder of this paper is organized as follows. In Section 2, we review the PIO and MPIO iteration methods, respectively. Furthermore, we also prove the overall convergence of the MPIO iteration method. Section 3 is devoted to the MMPIO iteration method, and its convergence property is proved. In Section 4, we discuss the choices of the parameters in the MMPIO iteration method, and give some heuristical strategies to choose appropriate parameters. In Section 5, Numerical examples on several PageRank problems are given to test the effectiveness of the MMPIO iteration method. Finally, we draw some conclusions in Section 6.

2. The PIO and MPIO iteration methods

In this section, we firstly review the PIO and MPIO iteration methods, respectively. Next, we prove that the MPIO iteration method converges linearly to the exact PageRank vector without any other restriction on its parameters.

2.1 The PIO iteration method [10]

Applying the power method to solve (1.2), we have the following iteration sequence:

$$x_{k+1} = \alpha P x_k + (1 - \alpha)v, \ k = 0, 1, 2, \cdots . \tag{2.1}$$

Gleich et al. [24] proposed an inner-outer iteration method for solving (1.2). At first, they reformulated (1.2) as

$$(I - \beta P)x = (\alpha - \beta)Px + (1 - \alpha)v, \tag{2.2}$$

with $0 < \beta < \alpha$, and obtained the following iteration sequence

$$(I - \beta P)x_{k+1} = (\alpha - \beta)Px_k + (1 - \alpha)v, \quad k = 0, 1, \cdots , \tag{2.3}$$

which is the so-called outer iteration.

In order to efficiently solve the linear system (2.3) with the coefficient matrix $I-\beta P$, an inner Richardson iteration is used to approximate x_{k+1}. First, setting the right-hand side of (2.3) as

$$f = (\alpha - \beta)Px_k + (1-\alpha)v,$$

and defining the following inner linear system:

$$(I - \beta P)y = f, \tag{2.4}$$

then (2.4) can be solved by the following inner iteration

$$y_{j+1} = \beta P y_j + f,\ j = 0, 1, 2, \cdots, l-1, \tag{2.5}$$

where y_0 is given by x_k as the initial guess and y_l is assigned as the new x_{k+1}.

For the outer iteration (2.3), it is stopped by

$$\|(1-\alpha)v - (I - \alpha P)x_{k+1}\|_1 < \tau,$$

and the inner iteration (2.5) is terminated by

$$\|f - (I - \beta P)y_{j+1}\|_1 < \eta,$$

where τ and η are the outer and inner tolerances, respectively.

Combining (2.1) with (2.3) and using the idea of two-step matrix splitting iteration, Gu et al. [10] proposed the following PIO iteration method:

The PIO iteration method:

$$\begin{cases} x_{k+\frac{1}{2}} = \alpha P x_k + (1-\alpha)v, \\ (I - \beta P)x_{k+1} = (\alpha - \beta)P x_{k+\frac{1}{2}} + (1-\alpha)v \end{cases} \tag{2.6}$$

with $0 < \beta < \alpha,\ 0 < \alpha < 1$ and $x_0 = v$ as the initial value. For the second iteration in (2.6), the inner-outer iteration (2.3)-(2.5) is used to get the approximate solution of x_{k+1}.

Theorem 2.1 [10]. The iteration matrix of the PIO iteration (2.6) is given by

$$R_{PIO} = \alpha(\alpha - \beta)(I - \beta P)^{-1}P^2,$$

and the modulus of its eigenvalues is bounded by

$$s = \frac{\alpha(\alpha - \beta)}{1 - \beta}$$

with $\beta \in (0, \alpha)$. Therefore, it holds that

$$\rho(R_{PIO}) \leq s < 1,$$

which implies that the PIO iteration method converges to the exact solution of (1.2) for any initial vector x_0.

2.2 The MPIO iteration method and its overall convergence

By applying the multi-step power method and the inner-outer iteration [24], the authors [3] presented the MPIO iteration method as follows:

The MPIO iteration method:

$$\begin{cases} x_{k+\frac{1}{m+1}} = \alpha P x_k + (1-\alpha)v, \\ x_{k+\frac{2}{m+1}} = \alpha P x_{k+\frac{1}{m+1}} + (1-\alpha)v, \\ \quad \vdots \\ x_{k+\frac{m}{m+1}} = \alpha P x_{k+\frac{m-1}{m+1}} + (1-\alpha)v, \\ (I-\beta P)x_{k+1} = (\alpha-\beta)P x_{k+\frac{m}{m+1}} + (1-\alpha)v, \end{cases} \tag{2.7}$$

where $0 < \beta < \alpha$ and $0 < \alpha < 1$, m ($m \geq 2$) is the iteration number of using the power method. If $m = 1$, then (2.7) reduces to the PIO iteration method (2.6).

Algorithm 1: The MPIO iteration method

Input: $P,\ \alpha,\ \beta,\ v,\ \tau,\ \eta,\ m(m \geq 2)$

Output: x

1: $x \leftarrow v$
2: $z \leftarrow Px$
3: **while** $\|\alpha z + (1-\alpha)v - x\|_1 \geq \tau$
4: **for i=1:m**
5: $x \leftarrow \alpha z + (1-\alpha)v$
6: $z \leftarrow Px$
7: **end**
8: $f \leftarrow (\alpha-\beta)z + (1-\alpha)v$
9: **repeat**
10: $x \leftarrow f + \beta z$
11: $z \leftarrow Px$
12: **until** $\|f + \beta z - x\| < \eta$
13: **end while**
14: $x \leftarrow \alpha z + (1-\alpha)v$

Theorem 2.2 [3]. The iteration matrix of the MPIO iteration (2.7) is expressed as

$$R_{MPIO} = \alpha^m(\alpha-\beta)(I-\beta P)^{-1}P^{m+1},$$

and the modulus of its eigenvalues is given by

$$\tilde{s} = \frac{\alpha^m(\alpha-\beta)}{1-\beta},\ 0 < \alpha < 1,\ 0 < \beta < \alpha.$$

Therefore, it holds that $\rho(R_{MPIO}) \leq \tilde{s} < 1$, i.e., the MPIO iteration method converges to the exact solution of (1.2) for any initial vector x_0.

In [29], the overall convergence of the inner-outer iteration method was given without imposing any restriction on the damping factors and the stopping tolerances, then the similar conclusion for the MPIO iteration method can also be obtained, as well as the PIO iteration method. In fact, the MPIO iteration method can be written as the following two-stage matrix splitting iteration framework [22]:

$$\begin{cases} x_{k,0} = x_k,\ x_{k+1} = x_{k,m_k}, \\ x_{k,j+1} = \beta P x_{k,j} + (\alpha-\beta)\alpha^m P^{m+1} x_k + (1-\alpha)((\alpha-\beta)\sum_{s=0}^{m-1}(\alpha P)^s P + I)v,\ k = 0,1,2,\cdots, \\ j = 0,1,2,\cdots, m_k - 1. \end{cases} \tag{2.8}$$

Theorem 2.3. Let $0 < \beta < \alpha$ and m_k be the number of the inner iteration steps at the $k-$th outer iteration with $m_k \geq 1$. Then the iteration sequence $\{x_k\}_{k=0}^{\infty}$ generated by (2.8) converges linearly to the exact PageRank vector x^*.

Proof. From (2.8), we have

$$x_{k,j+1} = \left((\beta P)^{j+1} + (\alpha-\beta)\alpha^m \sum_{t=0}^{j}(\beta P)^t P^{m+1}\right) x_k + (1-\alpha)\sum_{t=0}^{j}(\beta P)^t \left((\alpha-\beta)\sum_{s=0}^{m-1}(\alpha P)^s P + I\right) v.$$

Then it follows that

$$x_{k+1} = E_k x_k + F_k v, \quad k = 0,1,2,\cdots, \tag{2.9}$$

where

$$\begin{cases} E_k = (\beta P)^{m_k} + (\alpha-\beta)\alpha^m \sum_{t=0}^{m_k-1}(\beta P)^t P^{m+1}, \\ F_k = (1-\alpha)\sum_{t=0}^{m_k-1}(\beta P)^t \left((\alpha-\beta)\sum_{s=0}^{m-1}(\alpha P)^s P + I\right),\ k = 0,1,2,\cdots. \end{cases}$$

Since x^* is the exact PageRank vector, then from (2.9) we obtain

$$x^* = E_k x^* + F_k v, \quad k = 0,1,2,\cdots. \tag{2.10}$$

Subtracting (2.10) from (2.9), then

$$x_{k+1} - x^* = E_k(x_k - x^*) = \cdots = E_k E_{k-1}\cdots E_0(x_0 - x^*), \quad k = 0,1,2,\cdots$$

and

$$\begin{aligned} E_k &= (\beta P)^{m_k} + (\alpha-\beta)\alpha^m \sum_{t=0}^{m_k-1}(\beta P)^t P^{m+1} \\ &= (\beta P)^{m_k} + \alpha^m \sum_{t=0}^{m_k-1}(\beta P)^t[(I-\beta P) - (I-\alpha P)]P^m \\ &= (\beta P)^{m_k} + (\alpha P)^m (I - (\beta P)^{m_k}) - (\alpha P)^m \sum_{t=0}^{m_k-1}(\beta P)^t (I - \alpha P). \end{aligned} \tag{2.11}$$

Since $e^T P = e^T$, then from (2.11) it is clear that

$$\begin{aligned} e^T E_k &= \beta^{m_k} e^T + (\alpha^m - \alpha^m\beta^{m_k})e^T - \alpha^m e^T \sum_{t=0}^{m_k-1}(\beta P)^t (I - \alpha P) \\ &= (\beta^{m_k} + \alpha^m - \alpha^m\beta^{m_k})e^T - \alpha^m(1-\alpha)\sum_{t=0}^{m_k-1}\beta^t e^T \\ &= \frac{(\beta^{m_k}+\alpha^m-\alpha^m\beta^{m_k})(1-\beta)-\alpha^m(1-\alpha)(1-\beta^{m_k})}{1-\beta} e^T \\ &= \frac{((1-\alpha^m)\beta^{m_k}+\alpha^m)(1-\beta)-\alpha^m(1-\alpha)(1-\beta^{m_k})}{1-\beta} e^T. \end{aligned} \tag{2.12}$$

Since E_k $(k = 0, 1, 2, \cdots)$ are nonnegative matrices and $0 < \beta < 1$, then from (2.12) we have

$$\begin{aligned} \|E_k\|_1 &= \tfrac{((1-\alpha^m)\beta^{m_k}+\alpha^m)(1-\beta)-\alpha^m(1-\alpha)(1-\beta^{m_k})}{1-\beta} \\ &< \tfrac{((1-\alpha^m)\beta^{m_k}+\alpha^m)(1-\beta)}{1-\beta} \\ &< \tfrac{(1-\alpha^m+\alpha^m)(1-\beta)}{1-\beta} = 1. \end{aligned}$$

Let $\chi = \max\limits_k\{\delta_k\} < 1$ $(k = 0, 1, 2, \cdots)$ with $\delta_k = \|E_k\|_1$. Then

$$\begin{aligned} \|x_{k+1} - x^*\|_1 &\le \|E_k E_{k-1} \cdots E_0\|_1 \|x_0 - x^*\|_1 \\ &\le \|E_k\|_1 \|E_{k-1}\|_1 \cdots \|E_0\|_1 \|x_0 - x^*\|_1 \\ &= \delta_k \delta_{k-1} \cdots \delta_0 \|x_0 - x^*\|_1 \\ &\le \chi^{k+1} \|x_0 - x^*\|_1. \end{aligned} \tag{2.13}$$

Hence, from (2.13) the iteration sequence generated by (2.8) converges to the exact PageRank vector x^* as $k \to \infty$, and the proof is completed. □

Let $m = 1$, the two-stage matrix splitting iteration frame of the PIO iteration (2.6) can be obtained from (2.8):

$$\begin{cases} x_{k,0} = x_k,\ x_{k+1} = x_{k,m_k}, \\ x_{k,j+1} = \beta P x_{k,j} + \alpha(\alpha - \beta)P^2 x_k + (1-\alpha)((\alpha-\beta)P + I)v, \\ k = 0, 1, 2, \cdots, j = 0, 1, 2, \cdots, m_k - 1. \end{cases} \tag{2.14}$$

Theorem 2.4. Assume that $0 < \beta < \alpha$ and m_k is the number of the inner iteration steps at the $k-$th outer iteration with $m_k \ge 1$. Then the iteration sequence $\{x_k\}_{k=0}^{\infty}$ derived from (2.14) converges to the exact PageRank vector x^*.

3. The MMPIO iteration method

Let

$$I - \alpha P = M - N \tag{3.1}$$

be a matrix splitting of $I - \alpha P$ and M be an invertible matrix. Then the iteration sequence based on (3.1) for solving the PageRank problem (1.1) is given by

$$M x_{k+1} = N x_k + (1-\alpha)v,\ k = 0, 1, 2, \cdots. \tag{3.2}$$

Substituting the multi-step power iterations by the multi-step iterations (3.2), then we obtain the following MMPIO iteration method:

$$\begin{cases} M x_{k+\frac{1}{m+1}} = N x_k + (1-\alpha)v, \\ M x_{k+\frac{2}{m+1}} = N x_{k+\frac{1}{m+1}} + (1-\alpha)v, \\ \quad \vdots \\ M x_{k+\frac{m}{m+1}} = N x_{k+\frac{m-1}{m+1}} + (1-\alpha)v, \\ (I - \beta P)x_{k+1} = (\alpha-\beta)P x_{k+\frac{m}{m+1}} + (1-\alpha)v. \end{cases} \tag{3.3}$$

If $M = I$ and $N = \alpha P$, then (3.3) becomes the MPIO iteration method.

Algorithm 2: The MMPIO iteration method

Input: $P,\ M,\ N,\ \alpha,\ \beta,\ v,\ \hat{\tau},\ m_k \geq 2,\ m\ (m \geq 2)$

Output: x

1: $x \leftarrow v$
2: $z \leftarrow Px$
3: **while** $\|\alpha z + (1-\alpha)v - x\|_1 \geq \hat{\tau}$
4: **for** i=1:m
5: $Mx \leftarrow Nx + (1-\alpha)v$
6: **end**
7: $z \leftarrow Px$
8: $f \leftarrow (\alpha - \beta)z + (1-\alpha)v$
9: **for** i=1:m_k
10: $x \leftarrow f + \beta z$
11: $z \leftarrow Px$
12: **end**
13: **end while**
14: $x \leftarrow \alpha z + (1-\alpha)v$

Lemma 3.1 [15]. Let $\|AB\| \leq \|A\| \cdot \|B\|$. Then $\|X\| < 1$ implies that $I - X$ is invertible, $(I-X)^{-1} = \sum_{i=0}^{\infty} X^i$, and $\|(I-X)^{-1}\| \leq \frac{1}{1-\|X\|}$.

Lemma 3.2. Let $I - \alpha P = M - N$ be a matrix splitting and $\rho(R) < \rho(\alpha P)$, where $R = M^{-1}N$. If $\{x_k\}_{k=0}^{\infty}$ are generated by the iteration sequence (3.2) and $\{y_k\}_{k=0}^{\infty}$ are derived from power iteration (2.1) for the same initial value x_0, then $\|R^m(x_k - x^*)\|_1 < \|(\alpha P)^m(y_k - x^*)\|_1$ for some k.

Proof. Since $\rho(R) < \rho(\alpha P)$, then it follows that

$$\|x_{k+m} - x^*\|_1 < \|y_{k+m} - x^*\|_1 \tag{3.4}$$

for some k. Since

$$\|x_{k+m} - x^*\|_1 = \|R^m(x_k - x^*)\|_1$$

and

$$\|y_{k+m} - x^*\|_1 = \|(\alpha P)^m(y_k - x^*)\|_1,$$

then from (3.4) we have

$$\|R^m(x_k - x^*)\|_1 < \|(\alpha P)^m(y_k - x^*)\|_1$$

and complete the proof. □

Theorem 3.1. Let $I - \alpha P = M - N$ be a matrix splitting. If $\rho(R) < \rho(\alpha P)$, then the iteration sequence $\{x_k\}_{k=0}^{\infty}$ generated by (3.3) converges to the exact PageRank vector x^* for any initial value x_0.

Proof. From (3.3), it is clear that

$$x_{k+1} = (\alpha - \beta)(I - \beta P)^{-1} P R^m x_k + (I - \beta P)^{-1} \left((1-\alpha)v + (\alpha - \beta) \sum_{i=0}^{m-1} P R^i c \right), \tag{3.5}$$

where $c = (1-\alpha)M^{-1}v$.

Since x^* is the exact PageRank vector, then from (3.3) we have

$$x^* = (\alpha-\beta)(I-\beta P)^{-1}PR^m x^* + (I-\beta P)^{-1}\left((1-\alpha)v + (\alpha-\beta)\sum_{i=0}^{m-1} PR^i c\right). \tag{3.6}$$

Subtracting (3.6) from (3.5), then

$$x_{k+1} - x^* = (\alpha-\beta)(I-\beta P)^{-1}PR^m(x_k - x^*). \tag{3.7}$$

From Lemmas 3.1, 3.2 and (3.7), we obtain

$$\begin{aligned}
\|x_{k+1}-x^*\|_1 &= \|(\alpha-\beta)(I-\beta P)^{-1}PR^m(x_k-x^*)\|_1 \\
&\le \|(\alpha-\beta)(I-\beta P)^{-1}P\|_1\|R^m(x_k-x^*)\|_1 \\
&< \|(\alpha-\beta)(I-\beta P)^{-1}P\|_1\|(\alpha P)^m(y_k-x^*)\|_1 \\
&\le \|(\alpha-\beta)(I-\beta P)^{-1}P\|_1\|(\alpha P)^m\|_1\|y_k-x^*\|_1 \\
&\le \tfrac{\alpha^m(\alpha-\beta)}{1-\beta}\|y_k-x^*\|_1 \\
&\le \psi\alpha^k\|y_0-x^*\|_1,
\end{aligned} \tag{3.8}$$

where $\psi = \frac{\alpha^m(\alpha-\beta)}{1-\beta}$ and $\|P\|_1 = 1$. It follows from (3.8) that $\|x_{k+1}-x^*\|_1 \to 0$ as $k \to \infty$, and the proof is completed. □

Let $m = 1$, The modified PIO iteration method can be derived from (3.3):

$$\begin{cases} Mx_{k+\frac{1}{2}} = Nx_k + (1-\alpha)v, \\ (I-\beta P)x_{k+1} = (\alpha-\beta)Px_{k+\frac{1}{2}} + (1-\alpha)v. \end{cases} \tag{3.9}$$

Theorem 3.2. If $I-\alpha P = M-N$ is a matrix splitting and $\rho(R) < \rho(\alpha P)$, then the iteration sequence $\{x_k\}_{k=0}^{\infty}$ obtained from (3.9) converges to the exact PageRank vector x^* for any initial vector x_0.

Proof. The proof is similar to that of Theorem 3.1. □

Let $P = D + L + U$, where D is the diagonal part of P, L is the strictly lower triangular part of P, and U is the strictly upper triangular part of P, respectively. Then the matrix splitting of the AOR iteration method [20] for solving (1.2) is

$$M_A = \frac{1}{\omega}(I-\alpha D-\gamma\alpha L),\ N_A = \frac{1}{\omega}((1-\omega)(I-\alpha D) + (\omega-\gamma)\alpha L + \omega\alpha U), \tag{3.10}$$

where $\omega,\ \gamma$ are two real parameters with $\omega \neq 0$. For different ω and γ, we can get the corresponding iteration methods from (3.10):

(1) Jacobi method: $\omega = 1,\ \gamma = 0$.

(2) Gauss-Seidel method: $\omega = 1,\ \gamma = 1$.

(3) SOR method: $\omega = \gamma$.

Since $I-\alpha P$ is a nonsingular $M-$matrix, then we give the following convergence theorem of the AOR method based on the matrix splitting (3.10) for solving (1.2).

Theorem 3.3. Let $G_A = M_A^{-1}N_A$ and $J = (I-\alpha D)^{-1}(\alpha L + \alpha U)$ be the AOR iteration matrix and Jacobi iteration matrix for solving (1.2), respectively. If $0 < \omega < \frac{2}{1+\rho(J)}$ and $0 \le \gamma \le \omega$, then

$$\rho(G_A) \le |1-\omega| + \omega\rho(J) < 1.$$

Proof. Let

$$\begin{aligned} T &= (I - \alpha D - \gamma\alpha L)^{-1}(|1-\omega|(I-\alpha D) + (\omega-\gamma)\alpha L + \omega\alpha U) \\ &= (I - \gamma\tilde{L})^{-1}(|1-\omega|I + (\omega-\gamma)\tilde{L} + \omega\tilde{U}), \end{aligned}$$

where $\tilde{L} = (I-\alpha D)^{-1}\alpha L$ and $\tilde{U} = (I-\alpha D)^{-1}\alpha U$, respectively. From Lemma 3.1 and $I-\alpha D > 0$, it follows that T is a nonnegative matrix [2]. Then by Theorem 2.7 [14] there exists an eigenvector $x \geq 0,\ x \neq 0$ such that

$$Tx = \rho(T)x,$$

i.e.,

$$(|1-\omega|I + (\omega-\gamma)\tilde{L} + \omega\tilde{U})x = \rho(T)(I-\gamma\tilde{L})x \tag{3.11}$$

Multiplying by $\frac{1}{\omega}$, then from(3.11) we have

$$\left(\frac{\omega-\gamma+\gamma\rho(T)}{\omega}\tilde{L} + \tilde{U}\right)x = \left(\frac{\rho(T)}{\omega} - \left|1-\frac{1}{\omega}\right|\right)x.$$

Since $\tilde{L} \geq 0$ and $\tilde{U} \geq 0$, it is clear that $\left(\frac{\omega-\gamma+\gamma\rho(T)}{\omega}\tilde{L} + \tilde{U}\right) \geq 0$, then we get

$$\frac{\rho(T)}{\omega} - \left|1-\frac{1}{\omega}\right| \leq \rho\left(\frac{\omega-\gamma+\gamma\rho(T)}{\omega}\tilde{L} + \tilde{U}\right).$$

If $\rho(T) \geq 1$, then

$$1 \leq \frac{\omega-\gamma+\gamma\rho(T)}{\omega} \leq \rho(T)$$

holds. Therefore,

$$\frac{\rho(T)}{\omega} - \left|1-\frac{1}{\omega}\right| \leq \frac{\omega-\gamma+\gamma\rho(T)}{\omega}\rho(\tilde{L}+\tilde{U}) \leq \rho(T)\rho(J) \tag{3.12}$$

with $\rho(J) = \rho(\tilde{L}+\tilde{U})$, then

$$\frac{\rho(T)}{\omega} - \left|1-\frac{1}{\omega}\right| \leq \rho(T)\rho(J) \tag{3.13}$$

Case 1: $\omega \leq 1$. For this case, we have

$$\frac{\rho(T)}{\omega} - \frac{1}{\omega} + 1 \leq \rho(T)\rho(J) < \rho(T)$$

and

$$\left(\frac{1}{\omega} - 1\right)(\rho(T) - 1) < 0,$$

which contradicts $\frac{1}{\omega} - 1 \geq 0$ and $\rho(T) \geq 1$.

Case 2: $\omega > 1$. From (3.13) we get

$$\frac{\rho(T)}{\omega} + \frac{1}{\omega} - 1 \leq \rho(T)\rho(J).$$

Since $\omega < \frac{2}{1+\rho(J)}$, it implies

$$\frac{1}{2}(1+\rho(J))(1+\rho(T)) - 1 < \rho(T)\rho(J)$$

and

$$(1-\rho(J))(\rho(T)-1)<0,$$

which also contradicts $1-\rho(J)>0$ and $\rho(T)-1\geq 0$.

From the above discussion we know that $\rho(T)<1$ holds. From (3.12) we have

$$\frac{\omega-\gamma+\gamma\rho(T)}{\omega}<1$$

and

$$\frac{\rho(T)}{\omega}-\left|1-\frac{1}{\omega}\right|\leq\rho(J),$$

then

$$\rho(T)\leq\omega\rho(J)+|1-\omega|. \tag{3.14}$$

Moreover, it is clear that $|G_A|\leq T$, then from Lemma 2.4 [14] we have

$$\rho(G_A)\leq\rho(T) \tag{3.15}$$

If $\omega\leq 1$, then

$$\omega\rho(J)+|1-\omega|=\omega\rho(J)+1-\omega<1. \tag{3.16}$$

While if $\omega>1$, we obtain

$$\begin{aligned}\omega\rho(J)+|1-\omega| &=\omega\rho(J)+\omega-1\\ &=\omega(\rho(J)+1)-1\\ &<\tfrac{2}{1+\rho(J)}(\rho(J)+1)-1\\ &=1.\end{aligned} \tag{3.17}$$

Then the proof is completed from (3.14)-(3.17). □

Theorem 3.4. Let $I-\alpha P=M-N$ be a matrix splitting. If $\|R\|_1<\alpha$ and $0<\beta<\alpha$, then the MMPIO iteration method based on the matrix splitting (M,N) converges faster than the MPIO iteration method.

Proof. From Theorem 2.2, it is clear that the iteration matrix of the MPIO iteration is

$$R_{MPIO}=\alpha^m(\alpha-\beta)(I-\beta P)^{-1}P^{m+1}$$

and

$$\rho(R_{MPIO})=\frac{\alpha^m(\alpha-\beta)}{1-\beta}.$$

According to (3.5), the iteration matrix of the MMPIO iteration is

$$R_{MMPIO}=(\alpha-\beta)(I-\beta P)^{-1}PR^m.$$

Then

$$\begin{aligned}\rho(R_{MMPIO}) &\leq\|(\alpha-\beta)(I-\beta P)^{-1}PR^m\|_1\\ &\leq\|(\alpha-\beta)(I-\beta P)^{-1}P\|_1\|R^m\|_1\\ &\leq\tfrac{\alpha-\beta}{1-\beta}\|R\|_1^m\\ &<\tfrac{\alpha^m(\alpha-\beta)}{1-\beta}\\ &=\rho(R_{MPIO})\end{aligned}$$

and the proof is completed. □

Theorem 3.5. Let $I - \alpha P = M - N$ be a matrix splitting and $R = M^{-1}N$. If $\|R\|_\chi < \alpha$ and $\|P\|_\chi \leq 1$ for some operator norm $\|\cdot\|_\chi$, then the MMPIO iteration method derived from the matrix splitting (M, N) is faster than the MPIO iteration method with $0 < \beta < \alpha$.

Proof. From Theorem 3.4, we have

$$\rho(R_{MPIO}) = \frac{\alpha^m(\alpha - \beta)}{1 - \beta}.$$

Since $0 < \beta < 1$ and $\|P\|_\chi \leq 1$, then $\|\beta P\|_\chi < 1$. From Lemma 3.1, it follows that

$$\|(I - \beta P)^{-1}\|_\chi \leq \frac{1}{1 - \beta\|P\|_\chi}.$$

Then

$$\begin{aligned}
\rho(R_{MMPIO}) &\leq \|(\alpha - \beta)(1 - \beta P)^{-1} P R^m\|_\chi \\
&\leq \|(\alpha - \beta)(1 - \beta P)^{-1} P\|_\chi \|R^m\|_\chi \\
&\leq (\alpha - \beta)\tfrac{\|P\|_\chi}{1 - \beta\|P\|_\chi}\|R\|_\chi^m \\
&\leq (\alpha - \beta)\tfrac{1}{1 - \beta}\|R\|_\chi^m \\
&< \tfrac{\alpha^m(\alpha - \beta)}{1 - \beta} \\
&= \rho(R_{MPIO})
\end{aligned}$$

and the proof is completed. □

Remark 1. If $\rho(R) < \rho(\alpha P)$, then from (2.7) and (3.3) we learn that $x_{k+\frac{m}{m+1}}$ obtained from (3.3) is closer to the exact PageRank vector x^* than $x_{k+\frac{m}{m+1}}$ generated by (2.7), so the MMPIO iteration method has more effectiveness than the MPIO iteration method for solving (1.2).

Remark 2. From Theorem 3.3, we can construct the MMPIO iteration method based on the AOR splitting (3.10). Furthermore, for different ω and γ in (3.10), the MMPIO iteration method with the corresponding splitting can be obtained, such as the Jacobi splitting, Gauss-Seidel splitting and SOR splitting, etc.

4. The choices of the parameters

In this section, we pay attention to the choices of the parameters in the MMPIO iteration method. From the numerical examples in Section 5, we find that the appropriate parameters can partly improve the convergence performance of the MMPIO iteration method in term of the iteration number and computational time, then it is essential to discuss the choices of the parameters, such as β and m_k, etc.

First, we discuss the choice of the parameter β. From (3.5), it follows that the iteration matrix of the MMPIO iteration method is

$$R_{MMPIO} = (\alpha - \beta)(I - \beta P)^{-1} P R^m. \tag{4.1}$$

Then

$$\rho(R_{MMPIO}) \leq \frac{\alpha - \beta}{1 - \beta}\|R^m\|_1.$$

Let $f(\beta) = \frac{\alpha-\beta}{1-\beta}$, by simple calculation, we have

$$f'(\beta) = \frac{\alpha - 1}{(1-\beta)^2} < 0$$

with $0 < \alpha < 1$. Then $f(\beta)$ is monotonically decreasing, and $f(\beta)$ is smaller for a larger β. However, from the analysis in [24], we know that the outer iterations (2.3) converge faster if β is close to α, but the inner iterations (2.5) converge faster if β is close to zero. Then $\beta \in [0.5, \alpha)$ is an appropriate choice, and $\beta = 0.5$ is adopted in our numerical examples in Section 5. We emphasize that, an appropriate parameter β only reduces the upper bound of the spectral radius of the iteration matrix (4.1), but does not decrease the spectral radius itself. However, the choices of parameter $\beta \in [0.5, \alpha)$ can achieve better numerical results, which is illustrated clearly in Section 5.

Next, for the parameter m_k in Algorithm 2, just as the analysis of η in [24], which is also true for m_k. A larger m_k may result in spending a long computational time performing inner iterations (2.5), just to compute a single outer iteration (2.3) at a time, and slow the overall convergence. For a smaller m_k, on the other hand, may lead to inner iterations (2.5) that do not sufficiently approximate the exact solution of (2.4), then do not yield sufficient progress for the exact PageRank vector, and $m_k = 2$ or 3 may be an appropriate choice .

5. Numerical results

In this section, we illustrate the effectiveness of the MMPIO iteration compared with the PIO iteration and the MPIO iteration, respectively. The numerical experiments are performed in Matlab R2010 on an Intel dual core processor (2.30 GHz, 8GB RAM). We use four iteration parameters to test these iteration methods, which are the iteration step (denoted as IT), the computing time in seconds (denoted as CPU), the number of matrix-vectors (denoted as MV), and the relative residual (denoted as RES) defined as $\frac{\|r_k\|_2}{\|(1-\alpha)v\|_2}$ with $r_k = (1-\alpha)v - (I - \alpha P)x_k$.

Five test matrices P are listed in Table 1, where "Average Nonzeros" means the average number of the nonzero elements per row. All the data files are available from [27]. For the sake of justice, we take the teleportation vector $x_0 = v$ as the initial guess for all the test matrices. All algorithms are terminated once the residual norms RES$< 10^{-8}$. The damping factors are chosen as $\alpha = 0.85,\ 0.90,\ 0.95,\ 0.99$ in all numerical experiments.

Table 1: Five test matrices for (1.2).

Name	Size	Nonzeros	Average Nonzeros
Minnesota	2,642× 2,642	6,606	2.50
Wb-cs-stanford	9,914×9,914	36,854	3.71
Usroads	129,164×129,164	330,870	2.56
Flickr	820,878×820,878	9,837,214	1.45×10^{-5}
Wikipedia-20051105	1,634,989×1,634,989	19,753,078	7.38×10^{-6}

Example 1. In this example, we compare the MMPIO iteration method based on (3.10) with the MPIO iteration method, where we choose $\beta = 0.5$ and $m_k = 2$. The test matrix is the Minnesota matrix.

Numerical results are listed in Table 2. From table 2, we notice that the MMPIO iteration method with $\omega = 1.2$ and $\gamma = 1.1$ performs better than the MPIO iteration method in both iteration number and CPU time for different values of m, which is more obvious when α tends to 1. Fig.1 depicts the convergence curves of these algorithms with $\omega = \gamma = 1.2$. It shows that the MMPIO iteration method with the SOR splitting converges faster than the MPIO iteration method for different values of m.

Just as the MPIO iteration method, the effectiveness of the MMPIO iteration method is also parameter-dependent. For $\alpha = 0.85, 0.90$ in Table 2, the number of matrix-vectors and CPU time firstly decrease with the choices of m increasing, then they begin to increase, such as $m = 7, 10$.

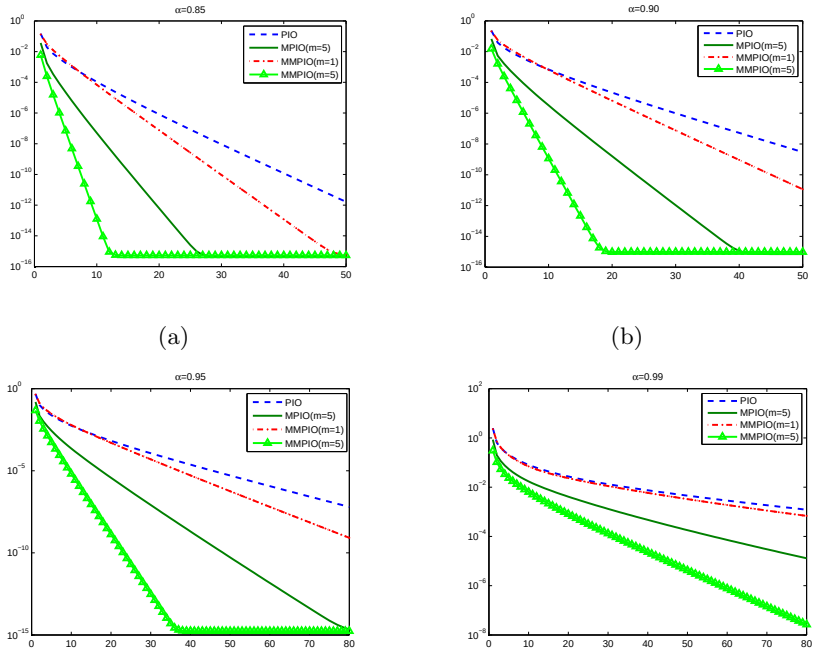


Figure 1: Convergence curves for the Minnesota matrix

Example 2. This example is devoted to the convergence performance of the MMPIO iteration method for different choices of the parameters β and m_k, respectively. The test matrix is the Wb-cs-stanford matrix.

Fig. 2 shows the iteration number of the MMPIO iteration method for different values of β, where we choose $m_k = 2$, $\omega = \gamma = 1.2$ and $m = 1, 3, 5, 7$, respectively. From Fig. 2, it follows that the MMPIO iteration method converges faster for a larger β, which is more evident for a smaller m, such as $m = 1$. For larger m, we observe that the iteration count does not change dramatically for $\beta \geq 0.5$, then it is appropriate to choose the values of β in the interval $[0.5, \alpha)$, which is consistent with the discussion in Section 4.

The numerical results for the MMPIO iteration method with different m_k are reported in

Table 2: Numerical results for the Minnesota matrix.

		MPIO			MMPIO		
α	m	IT(MV)	CPU	RES	IT(MV)	CPU	RES
	m=1	30(120)	0.7633	9.18×10^{-9}	28(68)	0.4372	4.11×10^{-9}
	m=3	17(102)	0.6003	6.68×10^{-9}	8(48)	0.3077	2.65×10^{-9}
0.85	m=5	12(96)	0.5473	5.02×10^{-9}	6(48)	0.2845	5.88×10^{-9}
	m=7	9(90)	0.5036	6.47×10^{-9}	5(50)	0.2843	1.91×10^{-9}
	m=10	7(91)	0.5079	3.16×10^{-9}	4(52)	0.2908	2.70×10^{-9}
	m=1	46(184)	1.1753	9.36×10^{-9}	24(96)	0.6202	9.61×10^{-9}
0.90	m=3	26(156)	0.9157	7.19×10^{-9}	9(54)	0.3286	3.15×10^{-9}
	m=5	18(144)	0.8439	7.10×10^{-9}	6(48)	0.2810	3.51×10^{-9}
	m=7	14(140)	0.7752	5.60×10^{-9}	5(50)	0.2877	8.92×10^{-9}
	m=10	10(130)	0.7270	8.82×10^{-9}	4(52)	0.3205	4.99×10^{-9}
	m=1	93(372)	2.3589	9.81×10^{-9}	48(192)	1.2134	7.71×10^{-9}
0.95	m=3	52(312)	1.8462	8.85×10^{-9}	15(90)	0.5379	5.63×10^{-9}
	m=5	36(288)	1.6340	8.80×10^{-9}	9(72)	0.4147	5.28×10^{-9}
	m=7	28(280)	1.5580	6.98×10^{-9}	7(70)	0.3878	1.40×10^{-9}
	m=10	21(273)	1.4808	5.70×10^{-9}	5(65)	0.3609	8.93×10^{-9}
	m=1	443(1772)	11.235	9.97×10^{-9}	225(900)	5.7270	9.85×10^{-9}
0.99	m=3	247(1482)	8.7218	9.48×10^{-9}	75(450)	2.6520	8.69×10^{-9}
	m=5	171(1368)	7.7608	9.47×10^{-9}	45(360)	2.0430	8.67×10^{-9}
	m=7	131(1310)	7.2132	9.24×10^{-9}	33(330)	1.8401	6.48×10^{-9}
	m=10	97(1261)	6.9041	9.01×10^{-9}	23(299)	1.6288	6.98×10^{-9}

Table 3, where $\beta = 0.5$, $\omega = \gamma = 1.2$ and $m = 2$, respectively. From Table 3, it is clear that the MMPIO iteration method needs less iteration number for a larger m_k. However, it also needs more CPU time and the number of matrix-vectors, for example, the case $m_k = 2$ compared with that of $m_k = 5$. Thus, $m_k = 2, 3$ may be good choices for the MMPIO iteration method, which is in concord with our conclusions in Section 4.

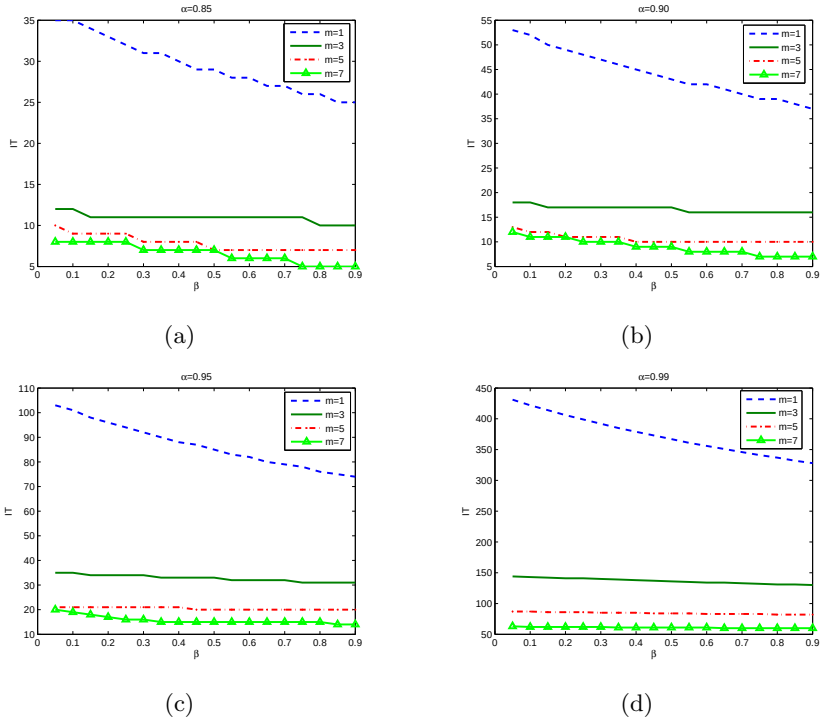


Figure 2: The iteration number for the Wb-cs-stanford matrix with different β.

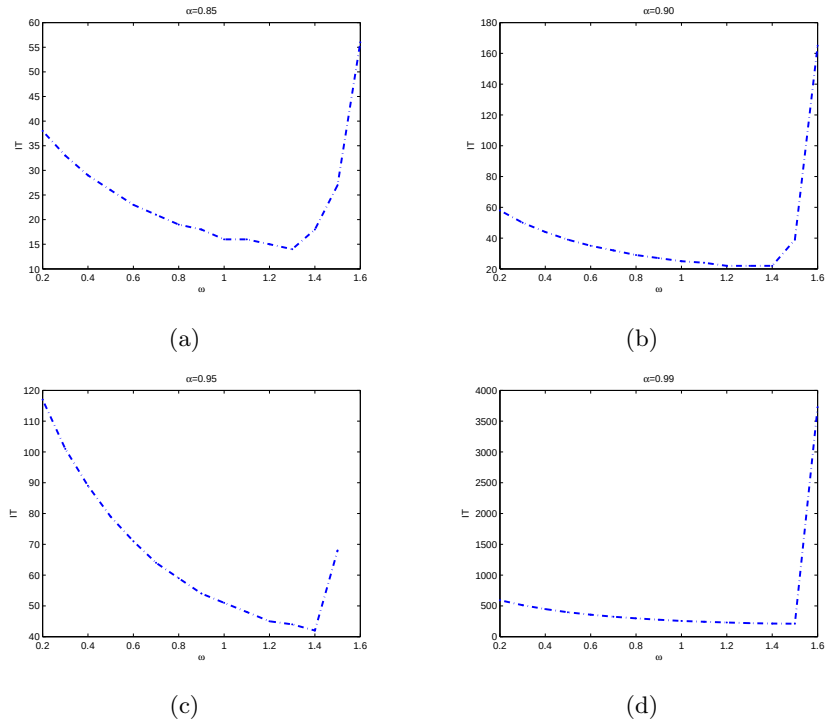
Example 3. In this example, we compare the convergence performance of the MMPIO iteration method with the MPIO iteration method for large test matrices, and discuss the choice of the parameter ω in (3.10). The test matrices are the Usroads, Flickr and Wikipedia-20051105 matrices, respectively.

First, by setting $\beta = 0.5$, $\omega = 1.2$, $\gamma = 0$ and $m_k = 2$, we make a comparison between the MMPIO iteration method and the MPIO iteration method for the Flickr and Wikipedia-20051105 matrices, respectively. The numerical results are listed in Tables 4, 5, from which we find that the MMPIO iteration method has more effectiveness than the MPIO iteration method in terms of iteration number and CPU time for different m, the advantage is more obvious for larger α, especially for the case $\alpha = 0.99$ with $m = 3$ in Table 5.

Next, we discuss the choice of the parameters ω in (3.10) for the MMPIO iteration method with $\beta = 0.5, m_k = 2, \gamma = 0$ and $m = 3$. The test matrix is the Usroads matrix. The numerical results are listed in Table 6 and Fig. 3, from which we observe that the MMPIO iteration method converges faster for $\omega > 1$, the choices of the parameter ω near 1.4 are satisfactory in this example.

Table 3: Numerical results for the Wb-cs-stanford matrix with different m_k.

		$m_k = 2$	$m_k = 3$	$m_k = 4$	$m_k = 5$
	IT(MV)	16(80)	15(90)	15(105)	15(120)
$\alpha = 0.85$	CPU	0.0268	0.0274	0.0297	0.0308
	RES	3.06×10^{-9}	6.95×10^{-9}	5.37×10^{-9}	4.91×10^{-9}
	IT(MV)	24(120)	23(138)	23(161)	23(184)
$\alpha = 0.90$	CPU	0.0477	0.0430	0.0428	0.0461
	RES	4.82×10^{-9}	6.95×10^{-9}	5.22×10^{-9}	4.69×10^{-9}
	IT(MV)	47(235)	45(270)	45(315)	44(352)
$\alpha = 0.95$	CPU	0.0745	0.0791	0.0897	0.0829
	RES	6.96×10^{-9}	9.80×10^{-9}	7.18×10^{-9}	9.58×10^{-9}
	IT(MV)	196(980)	190(1140)	187(1309)	186(1488)
$\alpha = 0.99$	CPU	0.3076	0.3111	0.3281	0.3594
	RES	9.24×10^{-9}	9.80×10^{-9}	9.78×10^{-9}	9.48×10^{-9}



(a) (b) (c) (d)

Figure 3: The iteration number for the Usroads matrix with different ω.

Table 4: Numerical results for the Flickr matrix.

		MPIO			MMPIO		
α	m	IT(MV)	CPU	RES	IT(MV)	CPU	RES
0.85	m=1	40(160)	9.5510	7.03×10^{-9}	36(144)	8.7198	9.92×10^{-9}
	m=3	22(132)	7.4372	7.84×10^{-9}	19(114)	6.5285	8.79×10^{-9}
	m=5	16(128)	6.6620	2.93×10^{-9}	13(104)	5.7535	7.33×10^{-9}
0.90	m=1	58(232)	14.0082	7.75×10^{-9}	53(212)	13.0407	8.75×10^{-9}
	m=3	32(192)	10.5722	8.36×10^{-9}	28(168)	9.5482	8.02×10^{-9}
	m=5	22(176)	9.2301	9.35×10^{-9}	19(152)	8.2301	7.98×10^{-9}
0.95	m=1	100(400)	24.0234	9.70×10^{-9}	93(372)	22.6101	8.26×10^{-9}
	m=3	56(336)	18.3204	8.04×10^{-9}	49(294)	16.4515	8.42×10^{-9}
	m=5	39(312)	16.3757	7.07×10^{-9}	33(264)	14.4027	9.89×10^{-9}
0.99	m=1	345(1380)	82.2261	9.98×10^{-9}	320(1280)	77.1992	9.65×10^{-9}
	m=3	192(1152)	63.5606	9.70×10^{-9}	169(1014)	56.6707	9.87×10^{-9}
	m=5	133(1064)	55.8008	9.61×10^{-9}	115(920)	50.1393	9.74×10^{-9}

Table 5: Numerical results for the Wikipedia-20051105 matrix.

		MPIO			MMPIO		
α	m	IT(MV)	CPU	RES	IT(MV)	CPU	RES
	m=1	41(164)	45.1763	7.10×10^{-9}	33(132)	37.2843	7.00×10^{-9}
0.85	m=3	23(138)	34.8688	5.52×10^{-9}	17(102)	25.4535	6.38×10^{-9}
	m=5	16(128)	31.2787	4.91×10^{-9}	13(104)	25.1067	5.84×10^{-9}
	m=1	61(244)	65.7905	9.97×10^{-9}	50(200)	53.7981	8.01×10^{-9}
0.90	m=3	34(204)	50.9168	8.98×10^{-9}	26(156)	39.0796	6.43×10^{-9}
	m=5	24(192)	45.6513	6.21×10^{-9}	22(176)	41.8118	6.36×10^{-9}
	m=1	120(480)	131.5475	8.90×10^{-9}	100(400)	108.6591	9.66×10^{-9}
0.95	m=3	67(402)	99.5178	7.94×10^{-9}	51(306)	75.6271	7.38×10^{-9}
	m=5	46(368)	88.8287	9.14×10^{-9}	37(296)	70.8631	6.98×10^{-9}
	m=1	516(2064)	564.6483	9.80×10^{-9}	507(2028)	542.9624	9.88×10^{-9}
0.99	m=3	287(1722)	478.5987	9.57×10^{-9}	249(1494)	366.8775	9.51×10^{-9}
	m=5	235(1645)	390.3816	9.44×10^{-9}	206(1442)	344.0656	9.53×10^{-9}

Table 6: Numerical results for the Usroads matrix with different ω.

m=3	α	0.85	0.90	0.95	0.99
	Iter(MV)	33(198)	50(300)	101(606)	511(3066)
$\omega = 0.3$	CPU	0.4946	0.7765	1.5101	7.6060
	RES	8.32×10^{-9}	9.16×10^{-9}	9.91×10^{-9}	9.84×10^{-9}
	Iter(MV)	26(156)	39(234)	79(474)	397(2382)
$\omega = 0.5$	CPU	0.3931	0.5878	1.2188	5.9697
	RES	7.31×10^{-9}	9.09×10^{-9}	9.28×10^{-9}	9.99×10^{-9}
	Iter(MV)	21(126)	32(192)	64(384)	324(1944)
$\omega = 0.7$	CPU	0.3264	0.4841	0.9669	4.7504
	RES	7.53×10^{-9}	7.74×10^{-9}	9.82×10^{-9}	9.78×10^{-9}
	Iter(MV)	18(108)	27(162)	54(324)	275(1650)
$\omega = 0.9$	CPU	0.2801	0.4117	0.8359	4.0637
	RES	5.31×10^{-9}	7.37×10^{-9}	9.98×10^{-9}	9.56×10^{-9}
	Iter(MV)	16(96)	25(150)	51(306)	256(1536)
$\omega = 1.0$	CPU	0.2428	0.3758	0.7350	3.7157
	RES	9.91×10^{-9}	7.85×10^{-9}	8.24×10^{-9}	9.96×10^{-9}
	Iter(MV)	15(90)	22(132)	45(270)	230(1380)
$\omega = 1.2$	CPU	0.2304	0.3379	0.6798	3.4513
	RES	4.61×10^{-9}	9.60×10^{-9}	9.89×10^{-9}	9.37×10^{-9}
	Iter(MV)	18(108)	22(132)	42(252)	213(1278)
$\omega = 1.4$	CPU	0.2714	0.3372	0.6332	3.1470
	RES	4.26×10^{-9}	9.45×10^{-9}	9.96×10^{-9}	9.51×10^{-9}

6. Conclusion

In this paper, we give an MMPIO iteration method for solving the PageRank problem, in which we use the multi-step matrix splitting iteration instead of the power method. Numerical results on several PageRank problems verify that the MMPIO iteration method is superior to the MPIO and PIO iteration methods, respectively. However, our proposed method is rather parameter-dependent, hence how to determine the optimal parameters for the MMPIO iteration method is a problem worth researching in the future work.

Acknowledgements

The work is supported by the China Scholarship Council (201706935029).

References

[1] P. Boldi, M. Santini, S. Vigna, PageRank as a function of the damping factor,in:Proceedings of the 14th International World Web Conference, ACM, NewYork, 2005.

[2] A. Berman, R.J. Plemmons, Nonnegative Matrices in the Mathematical Sciences, Academic Press, NewYork, 1979.

[3] C. Wen, T.Z. Huang, Z.L. Shen, A note on the two-step matrix splitting iteration for computing PageRank, J. Comput. Appl. Math. 315 (2017) 87-97.

[4] G.H. Golub, C. Greif, An Arnoldi-type algorithm for computing PageRank, BIT Numerical Mathematics. 46 (2006) 759-771.

[5] L. Page, S. Brin, R. Motwami, T. Winograd, The Pagerank citation ranking: bringing order to the web, Technical Report, Computer Science Department, Stanford University, 1998.

[6] C.Q. Gu, L. Wang, On the multi-splitting iteration method for computing PageRank, J.Appl. Math.Comput. 42 (2013) 479-490.

[7] N. Huang, C.F. Ma, Parallel multisplitting iteration methods based on M-splitting for the PageRank problem, Appl. Math. Comput. 271 (2015) 337-343.

[8] S. Kamvar, T. Haveliwala, C. Manning, G. Golub, Extrapolation methods for accelerating PageRank computations,in:Proceedings of the 12th International World Web Conference, ACM, NewYork, 2003.

[9] T.H. Haveliwala, S.D. Kamvar, D. Klein, C. Manning, G.H. Golub, Computing PageRank using power extrapolation, Stanford University Technical Report, 2003.

[10] C.Q. Gu, F. Xie, K. Zhang, A two-step matrix splitting iteration for computing PageRank, J. Comput. Appl. Math. 278 (2015) 19-28.

[11] S.D. Kamvar, T.H. Haveliwala, G.H. Golub, Extrapolation methods for accelerating PageRank computations, Technique Report SCCM 03-02, Stanford University, 2003.

[12] G.H. Golub, C.F. Van Loan, Matrix Computations, third ed., The Johns Hopkins University Press, Baltimore, London, 1996.

[13] B.Y. Pu, T.Z. Huang, C. Wen, A preconditioned and extrapolation-accelerated GMRES method for PageRank, Appl. Math. Lett. 37 (2014) 95-100.

[14] R.S. Varga, Matrix Iterative Analysis, Springer-Verlag, Berlin Heidelberg, 2000.

[15] J.W. Demmel, Applied Numerical Linear Algebra, Society for Industrial and Applied Mathematics, Philadelphia, 1997.

[16] X.Y. Tan, A new extrapolation method for PageRank computations, J. Comput. Appl. Math. 313 (2017) 383-392.

[17] A. Arasu, J. Novak, A. Tomkins, J. Tomlin, PageRank computation and the structure of the web: experiments and algorithms, in:Proceedings of 11th International World Web Conference, Honolulu, 2002.

[18] C.Q. Gu,W.W. Wang, An Arnoldi-Inout algorithm for computing PageRank problems, J. Comput. Appl. Math. 309 (2017) 219-229.

[19] R. Morgan, M. Zeng, A harmonic restarted Arnoldi algorithm for calculating eigenvalues and determining multiplicity, Linear Algebra Appl. 415 (2006) 96-113.

[20] A. Hadjimos, Accelerated overrelaxation method, Math.Comp. 32 (1978) 149-157.

[21] M. Bianchini, M. Gori, F. Scarselli, Inside PageRank, ACMTrans. Internet Technol. 5 (2005) 92-128.

[22] Z.Z. Bai, J.C. Sun, D.R. Wang, A unified framework for the construction of various matrix multisplitting iterative methods for large sparse system of linear equations, Comput. Math. Appl. 32 (1996) 51-76.

[23] Z.X. Jia, Refined iterative algorithms based on Arnoldis process for large unsymmetric eigenproblems, Linear Algebra Appl. 259 (1997) 1-23.

[24] D.F. Gleich, A.P. Gray, C. Greif, T. Lau, An inner-outer iteration method for computing PageRank, SIAM J. Sci. Comput. 32 (2010) 349-371.

[25] A.N. Langville, C.D. Meyer, Googles PageRank and Beyond: The Science of Search Engine Rankings, Princet on University Press, Princeton, NJ, 2006.

[26] G. Wu, Y.M. Wei, A power-Arnoldi algorithm for computing PageRank, Numer. Linear Algebra Appl. 14 (2007) 521-546.

[27] http://www.cise.ufl.edu/research/sparse/matrices/Gleich/index.html.

[28] G. Wu, Y.M. Wei, An Arnoldi-extrapolation algorithm for computing PageRank, J. Comput. Appl. Math. 234 (2010) 3196-3212.

[29] Z.Z. Bai, On convergence of the inner-outer iteration method for computing PageRank, Numer. Algebra Control Optim. 2 (2012) 855-862.